Report No. 429

COO-2118-0003

SHOW-and-TELL

An Interactive Programming System For Image Processing

System Specifications

by

John Read

February 18, 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

'COO 2118-0003

Report No. 429

SHOW-AND-TELL
An Interactive Programming System For Image Processing

System Specifications

by

John Read

February 18, 1971

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Supported in part by Contract AT(11-1)-2118 with the U.S. Atomic Energy Commission.

TABLE OF CONTENTS

1. SYSTEM OVERVIEW

1.1 System Objectives

The Show-and-Tell (S&T) programming system is a console-oriented software package providing a facile method of using the operational components of the Illiac III computer.

The system operates on the hardware depicted in figure 1. An overall documentation of the Illiac III computer system is contained in reference [3].

S&T is designed to support two different types of programming objectives corresponding to a well-known decomposition of image processing into two phases: a preprocessing and feature-extraction phase followed by an interpretation phase.

Some typical tasks included in the first phase are:

- Cleaning up the digitized image to reduce noise introduced by the transducer and by the digitization process.
- Normalization of the digitized image to eliminate variations inherent in the object being scanned.
- Location of occurrences of primitive features such as edges or lines.
- Encoding of subsets of the image for input to phase 2 processing.

The second phase operates on the first phase output, which is in a symbolic form, rather than as an array of gray-scale values. It seems clear, however, that a data path must also lead from phase 2 back to phase 1 to permit interpretation routines to direct the acquisition of information from the original image.

Show-and-Tell's design is oriented towards this model. On one hand, it provides facilities for the manual control of scanning hardware so that preprocessing routines can be tried out on a large number of test images. On the other hand, a link is provided to the IBM 360/75 with its large amounts of storage and high-level programming languages

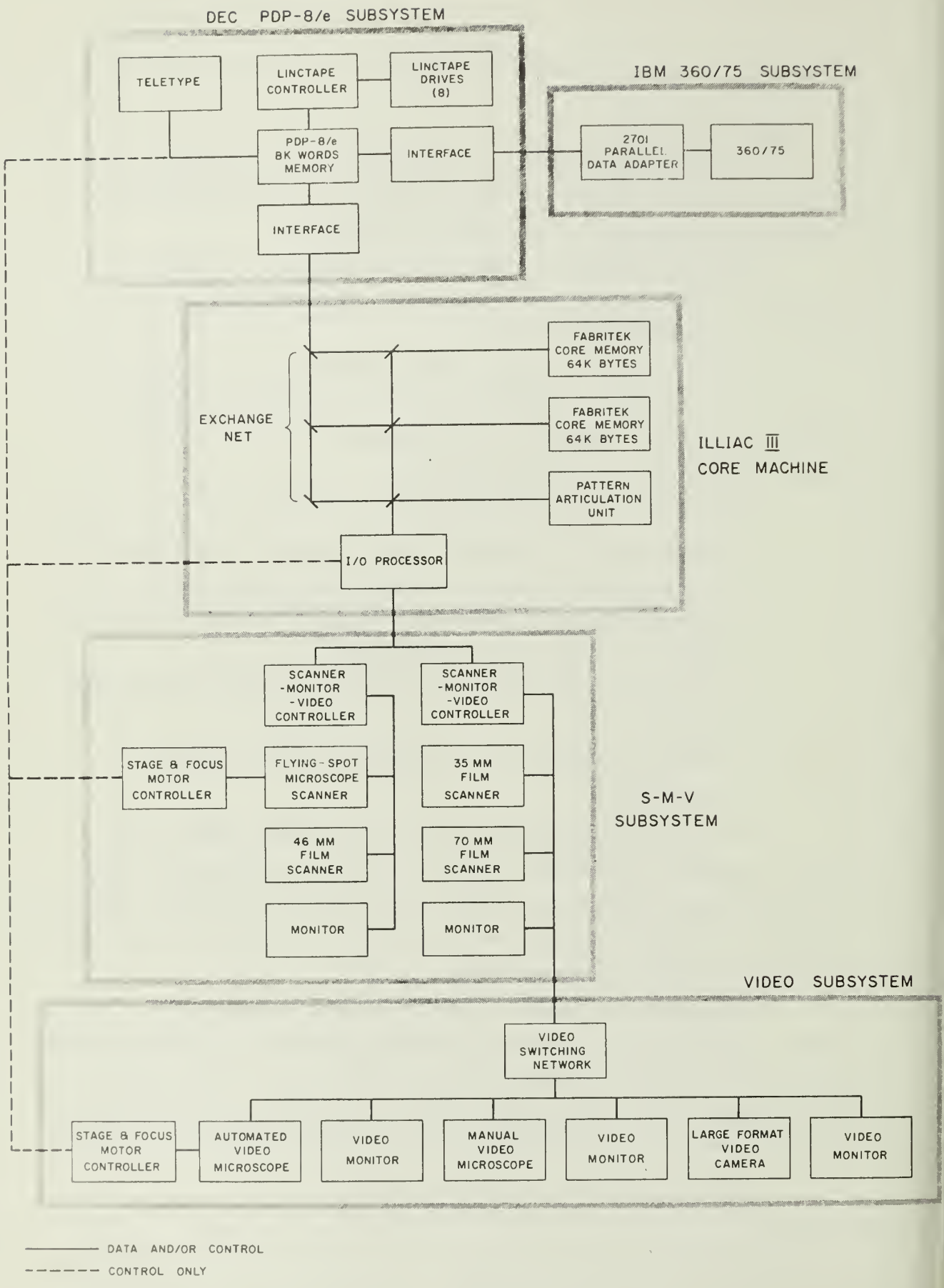


FIGURE 1 HARDWARE CONFIGURATION

so that phase 2's interpretation routines can be coded with a minimum of wasted effort and time. Furthermore, this link is bidirectional, in that interpretation programs written in a high-level language (currently FORTRAN/PAX) can provide feedback to the acquisition and preprocessing of phase 1.

Finally, due to the presently ill-defined requirements for picture-processing software, it was deemed wise to implement a minimal system at first, expecting that additions and changes would naturally occur as work on applications progresses. To this end, the system is fairly modular and includes provision for loading parts of the system into the very restricted (8K) PDP 8 core as they are required so as to reduce the need for tight or highly finished code.

1.2 System Elements

S&T is composed of the following subsystems:

- Translator: Converts S&T language typed by the operator into an interpretable list structure. Enough information is contained in this internal coding to reconstruct the source statements on demand for listing and editing.
- Interpreter: Operates on the aforementioned list structure and on the Data Storage Lists. (see below).
- Interpretable Code Area (ICA):
  Storage for the list structure produced by the Translator.
- Executable Code Area (ECA):
  Storage for programs executed directly by the PDP-8 hardware; i.e., the output of the PAL assembler.
- Interpretable Code Loader: Loads interpretable code into the ICA.

- Executable Code Loader: Loads executable code into the ECA.
- Supervisor: Provides first-level interrupt handling and controls loading of System Command Processors.
- System Command Processors: Carry out system commands. (see section 2).
- Data Storage Lists: Provides dynamically allocated storage for all operands. (see Section 3.2.2.5)

1.3 Use of the System

Figure 2 is a panorama of the Input/output room of the Illiac III system, with the general location of some of the I/O devices indicated. Figure 3 is a view of an operator station, showing a teletype for entering instructions and a monitor console. On the monitor screen, the result of executing a SCANM instruction can be seen. The SCANM instruction enables an operator to select a small (128 x 128 picture elements) picture segment for machine processing. The small, brightly-lighted square is the segment. A low-resolution, reasonably flicker-free representation of the entire picture is simultaneously available to give the operator a general idea of the location of points of interest.

One of the input devices, a high-resolution (1500 lines) video microscope is shown in figure 4. A variety of other picture input equipment, as shown schematically in figure 1, is also available or in various stages of completion.

All of the I/O devices are operated by a common control device, the Scanner-Monitor-Video Controller described in detail in reference [1]. This device permits program control of a wide selection of processing options, including a choice of raster size and location, sampling frequenc spot size, number of gray levels detected and some others.

Pictures are converted to digital form by the input devices and stored in a 128 K byte Fabritek core memory which can store 2^{20} bits of information. This capacity can be used in various ways, i.e., to store

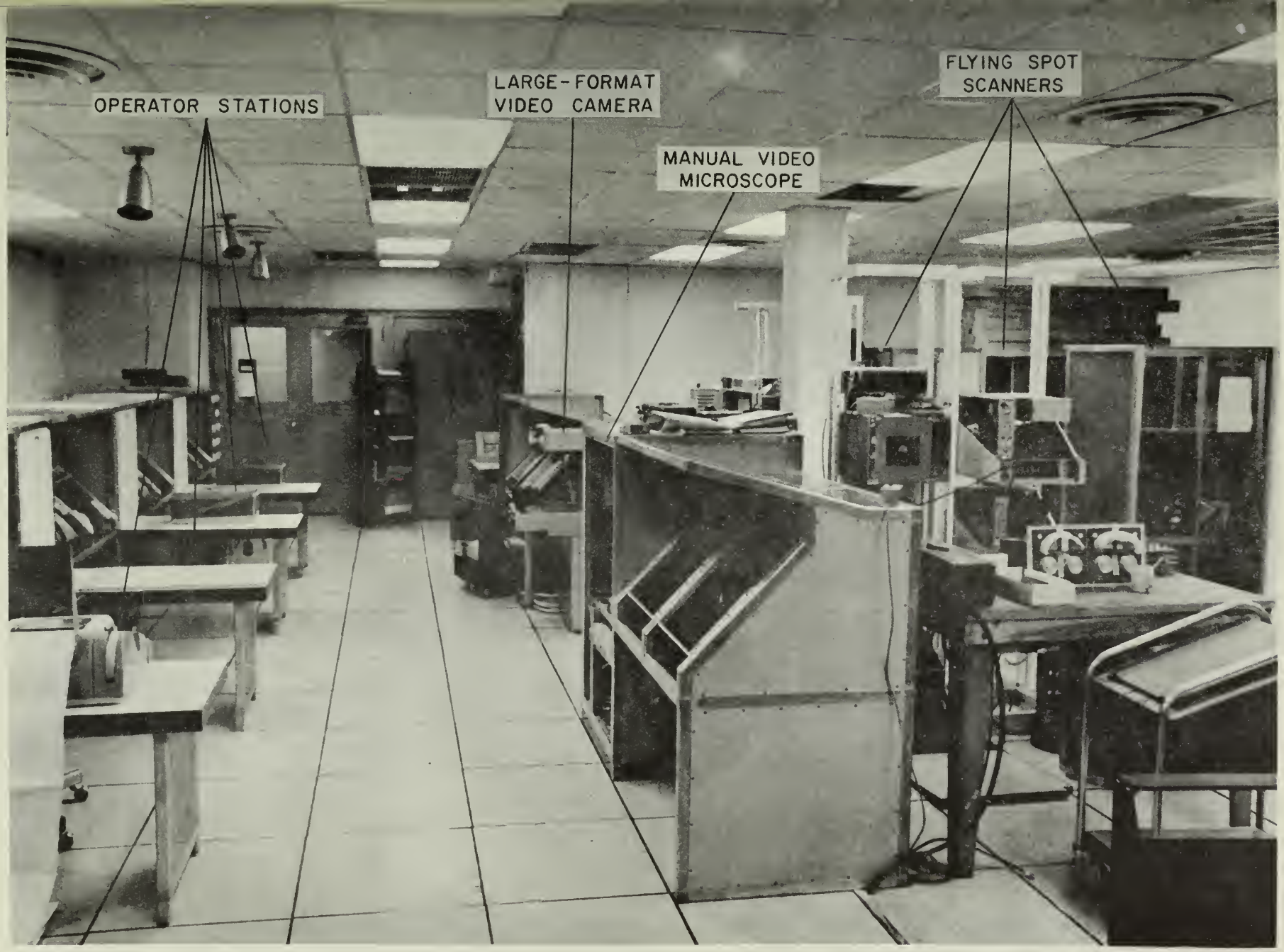

OPERATOR STATIONS
LARGE-FORMAT VIDEO CAMERA
MANUAL VIDEO MICROSCOPE
FLYING SPOT SCANNERS

Figure 2

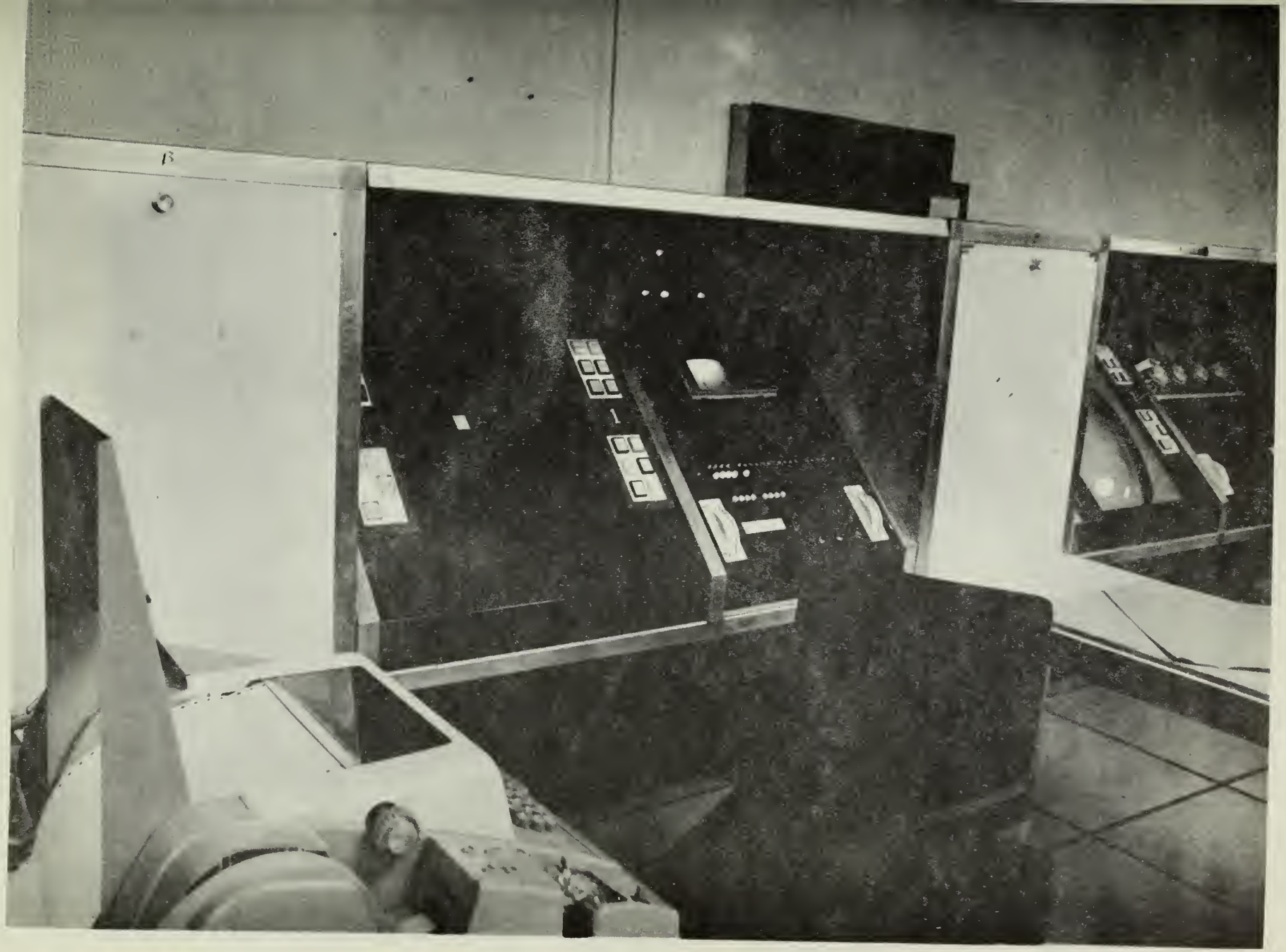


Figure 3

Figure 4

one picture of 512 x 512 picture elements (pixels) with four bits of gray-level data per pixel, or sixteen 128 x 128 pictures, or 64 one-bit x 128 x 128 planes, and so on.

The rest of this section contains a step-by-step description of the use of Show-and-Tell to develop a simple application, the counting of particles of a certain size and darkness.

In this sequence, the operator uses the XG (translate and go) command to learn a few useful facts about the image he wants to process, namely the size in pixels of a typical particle and a threshold value he can use to discriminate between particles and background. As each instruction is typed, it is translated and executed and the operator observes the effect. Having obtained the threshold and size numbers, the operator changes to translate mode (XL) and types a program to detect and count particles. He then executes the program, makes a change, re-executes it, and saves the program.

1. The operator identifies himself to S&T, (DEMO), wakes up the 360, (WK), and indicates that he is using the 46mm scanner (S46).

```
>LO                    1
CODE:DEMO
>WK
360 READY
>DV
CODE:S46
```

2. In order to measure some parameters of the image, the operator puts the system into translate-and-go mode, where each instruction is translated and executed immediately after it is typed (XG). He then selects a window and reads it into memory (SCANM). While selecting the image, he determines that less resolution is needed so he increases the x and y scanner sampling increment (P:++++ and Q:++++) to 2^4 in each case. The next step is to find out the size in pixels of the particles

1 Underlined characters were typed by Show-and-Tell

he wants to count, and also to establish a gray-scale threshold which can be used to discriminate between particles and background. To do this he executes a TRACE command which displays the picture he just read in with SCANM, and also a bright spot, or cursor, which he can move around on the picture. He uses this to point to a particle on the picture. S&T records the coordinates of the particle. The next instruction, PNABE, prints out the gray-scale values of those pixels in a 6 x 6 neighborhood of the point indicated, using a one-character-per-pixel notation, (such that a gray-scale value of ten is typed as an A). The operator notes that the pixels in the area of the particle generally have gray-scale value higher than four and that the particle has an area of seven pixels.

```
≥XG
   SCANM P.1
P:++++
O:++++
   ASGN N.1,1
   TRACE P.1,C.1,N.1
   PNABE P.1,C.1,6

211232
235671
157A82
212321
222111
222112
```

3. Having obtained these facts, the operator puts the system into translate mode (XL) and types in a program which will send a picture to the 360, (PICIN) and cause the PAX system on the 360 to put a one in picture P.2 at each pixel whose corresponding P.1 pixel had a gray-scale value greater than 5 (SLICE) and find and count all connected objects in P.2 with an area between 3 and 10. The count is then typed by the program. The program assumes that picture P.1 is already defined and has been read into core and that pictures P.2 through P.4 have been defined as planes.

```
>XL
000   ASGN N.2,1
001   ASGN N.1,0
002   ASGN I.1,((-1,-1),(-1,0),(1,-1),(1,0))
003   CALL PAX,'PICIN',P.1,4
004   CALL PAX,'SLICE',P.2,P.1,4,5,5
005 LOOP: CALL PAX,'XLISTX',P.2,C.1,N.2
006   CALL PAX,'CLEAR',P.3,0
007   CALL PAX,'XWRITC',P.3,C.1
008   CALL PAX,'XCONNE',I.1,P.4,P.2,P.3,N.2
009   CALL PAX,'AREA',P.4,0,0,N.2
010   CALL PAX,'AND',P.2,P.2,P.4,1
011   IFGO N.2,.GT,10,WRGSIZ
012   IFGO N.2,.LT,3,WRGSIZ
013   INCR N.1,1
014 WRGSIZ: CALL PAX,'NULL',P.2,N.3
015   IFGO N.3,.EQ,1,FINISH
016   GOTO LOOP
017 FINISH: ASGN T.1,N.1
018   TEXTO 'PARTICLE COUNT='
019   TEXTO T.1
020   EXIT
```

4. The program is then tested (GO), using the picture currently in Illiac III core. An answer is typed (PARTICLE COUNT =5) which is unsatisfactory. The operator decides to change (CH) the relation code in the SLICE instruction from a 5 to a 4, meaning a change from 'greater than' to 'greater than or equal to'. The program is re-executed, this time with better results.

```
≥GO
PARTICLE COUNT=5
≥CH
:4
004   CALL PAX,'SLICE',P.2,P.1,4,4,5
≥GO
PARTICLE COUNT=13
```

5. The corrected program is saved on the system device (SV) and given a file name of PCNT. The companion 360 program is terminated (KL) and the operator Logs off.

```
≥SV
*OUT-S:PCNT
≥KL
≥LF
```

2. SYSTEM COMMANDS

2.1 Entering System Commands

Execution of system programs such as the translator or the interpreter is controlled by typing system commands. S&T signals that it is ready to receive a system command by typing a 'greater than' symbol (>). S&T can be put into system command mode at any time by typing CTLE, i.e., holding down the CTL key on the teletype and pressing E. S&T will then type >.

2.2 Documentation of System Commands

In this section, underlined statements are those typed by the S&T system.

The syntax notation used in this report is almost the same as the standard IBM notation described in the front of, for example, the PL/1 Language Specifications Manual, reference [6]. The only differences are (1) for emphasis, BNF-type brackets are placed around "notation variables" as in <integer> and (2) BNF-format rewrite rules are used where this presentation seems clearer.

Command: CH

Purpose: To change statements in a program already in the Interpretable Code Area.

Execution: This command has the effect of a DL followed by an IN. A colon is typed. The operator responds with:

<statement number 1> [, <statement number 2>]

(see DL command). Statements entered are then translated and inserted in place of the ones just deleted. (See IN command).

Command: DL

Purpose: To delete code from the Interpretable Code Area.

Execution: A colon is typed. The operator's response is of the form:

<statement number 1> [, <statement number 2>]

If <statement number 2> is present, all statements between and including <statement number 1> and <statement number 2> are deleted. Otherwise, only the statement at <statement number 1> is deleted. If a labeled statement is deleted, the label must be placed on some other statement in the program. If this is not done, execution of the program will terminate if a GOTO, IFGO or CALL to that label is attempted.

Command: DV

Purpose: To identify which scanning device is to be used.

Execution: A request for a device code is typed:

CODE:

The operator selects the code from the following list and enters it.

S46 - 46 mm film scanner

SMS - Scanning microscope

VMS - Video microscope

VLF - Large format video

S70 - 70 mm film scanner

S35 - 35 mm film scanner

Command: GO

Purpose: To execute S&T code which has been loaded into the Interpretable Code Area by the Loader or the Translator.

Execution: The interpreter is invoked. Execution begins with the first location of the ICA. If the ICA is empty, an error message is typed and control returns to the System Command Processor.

Command: IN

Purpose: To insert code into a program already in the Interpretable Code Area.

Execution: A Colon is typed. The operator responds with the number of the statement before which the code is to be inserted. The Translator is then invoked and types a carriage return and line feed. All statements entered are translated and inserted. A CTLE statement terminates the command.

Command: KL

Purpose: To delete the PAXDRIVR task (see section 4, 'Interactive PAX').

Execution: A command is sent to PAXDRIVR which causes it to terminate.

Command: LS

Purpose: To print contents of Data Storage Lists currently at the top of the stack or the Interpretable Code Area.

Execution: A colon(:) is typed. The operator enters parameters describing the item to be listed:

<lower limit> [,<upper limit>]

Each limit is of the form

<data code> [.<integer>]

where<data code>is one of the Data Storage List Codes (except P), or is an S, meaning that the program in the ICA is to be printed. If .<integer> is omitted, the entire Data List or ICA is to be printed. <integer> for the ICA refers to a statement number.

Example: Typing N.3, N.10 causes the contents of counters three through ten to be printed. Another example: typing S.39 causes the 39th instruction in the ICA to be printed.

Command: LD

Purpose: To load previously translated and saved S&T code into the Interpretable Code Area.

Execution: The Loader is invoked. The PDP8 Disk Monitor is used to get the file Name from the Operator: * IN -

The operator types the device code and file name of the file containing the saved program. (See reference [4]). Note that since the interpreter always begins execution at the first location of the ICA, the main-line program must be loaded before subroutines.

Command: LF

Purpose: Updates accounting data.

Execution: Usage data is computed and entered into accounting tables,

---

Command: LØ

Purpose: Obtains a code from the operator to be used for accounting purposes.

Execution: Command is automatically typed when S&T is first entered:

> LØ

Code:

No other commands are accepted until the operator types a valid operator code.

Command: SV

Purpose: To save the contents of the Interpretable Code Area

Execution: The PDP8 Disk Monitor is used to get a file and device name from the operator: *OUT-
The operator types descriptors for the file to contain the ICA. (see reference [4]).

Command: WK

Purpose: To determine whether PAXDRIVR is active on the 360/75. (see section 5, 'Interactive PAX').

Execution: If PAXDRIVR has not started, the message 'WAITING FOR 360' is typed, and a WAIT loop is entered. When PAXDRIVR responds, '360 READY' is typed.

Command: XG

Purpose: Invokes the S&T Translator to translate one statement, then invokes the Interpreter to execute the instruction.

Execution: When the translator is ready, it types a carriage return-line feed. The statement is entered by the operator, translated and interpreted. As each instruction is translated, it overlays the previous one. Typing a CTLE returns control to the system command processor.

Command: XL

Purpose: To invoke the S&T Translator

Execution: When the S&T Translator is ready, it types a carriage return followed by a line feed. Statements may then be entered for translation. The Translator continues to accept statements until a CTLE is typed. Line numbers are automatically generated.

3. S&T LANGUAGE

3.1 Introduction

An interactive, incremental translator is provided for SHOW-AND-TELL, a language for processing pictures using the S-M-V and PAU subsystems of the Illiac III. The translator is interactive in that programs can be composed and debugged from a console typewriter and is incremental in that each statement is translated as it is entered and may be immediately executed, at the programmer's option. Alternatively, blocks of code may be translated *en masse*, saved, and called by the operator or by other programs.

The operations in the language are classified: PAU instructions, sequence control instructions, data-manipulating instructions and input-output instructions. PAU instructions are those to be executed by the Pattern Articulation Unit of the Illiac III.

To simplify the system, any operand to be processed is stored in one of eight predefined lists (see Sec. 3.2.2.5), with each list dedicated to a single type of data.

3.2 Language Elements

3.2.1 Statement Syntax

[<label>:] <instruction> CR

where CR is a carriage return. Statements may be continued by typing **LF** (line feed) instead of CR and continuing in the first column of the next line.

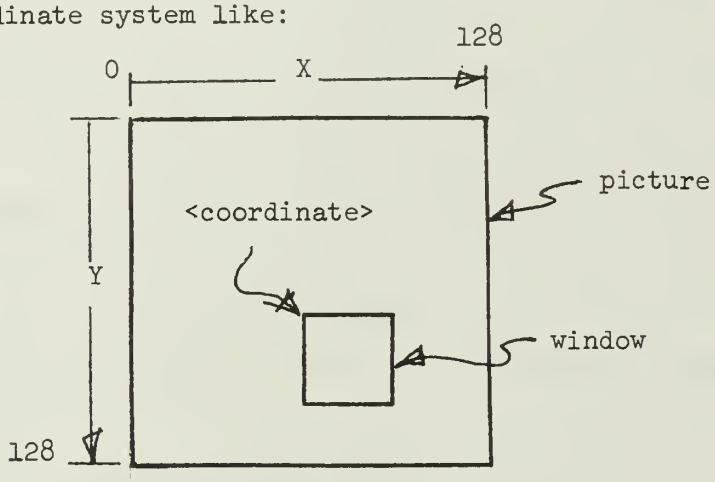
3.2.2 Instructions

Documentation of instructions is classified by instruction type. Operands are documented separately in section 3.2.3. At various places in the syntax, a symbol < * n *> appears, where n is a positive integer. These refer to notes which modify or clarify certain aspects of the syntax. The notes corresponding to the integers are in section 3.2.4.

3.2.2.1 PAU Instructions

References [3] and [5] describe the operation and programming of the PAU in depth; the documentation which follows describes syntax to use in writing instructions and does not repeat any of that information.

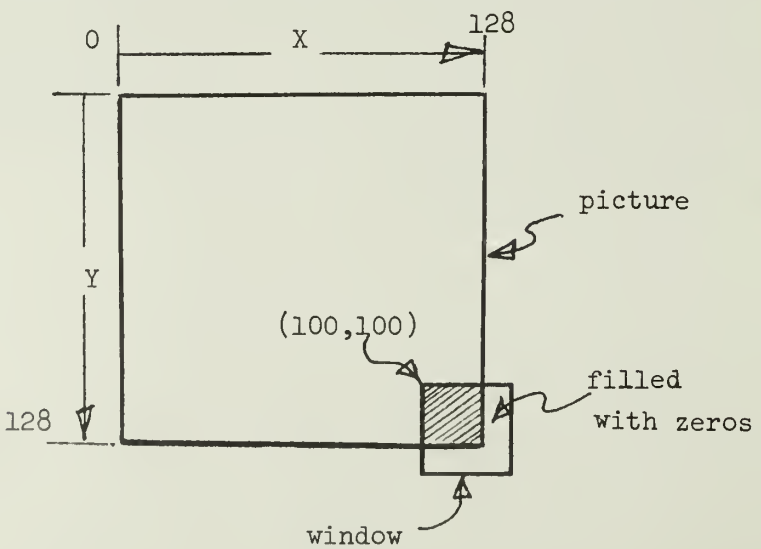
A few changes have been made in some of the instructions. STOREB and LOADB have been replaced by STOREW and LOADW. These instructions cause a 32 x 32 window with "upper left" corner at < coordinate > within picture P. <index> to be loaded in or stored from the IA, where "upper left" is used with reference to a coordinate system like:



If the edge of picture P. <index> falls within the window, then the window is truncated and the IA is filled out with zeroes. For example, if

<coordinate> = (100,100),

then the following situation obtains for a 128 x 128 picture:



PAU Instruction Syntax:

```
BOOLE     <bit-string reference>, <function reference>  <*S1*>

CLEARP    [<bit-string reference>], <plane reference>  <*S2*>

CONNEC    <plane reference>, <plane reference>,
          <plane reference>, <bit-string reference>  <*S3*>

LIST      <plane reference>, N. <index>,  C. <index>

COPY      <plane reference>, <plane reference>

COPYC     <plane reference>, <plane reference>

TALLY     [<plane reference>] [,<bit-string reference>] <*S3*> <*S4*>

TALLYH    [<plane reference>] [,<bit-string reference>] <*S3*> <*S4*>

SHIFT     <plane reference>, <displacement>

SETP      <plane reference>,[ <bit-string reference>]<*S2*>
```

PLAND <plane reference>, <plane reference>

PLOR <plane reference>, <plane reference>

PLEXOR <plane reference>, <plane reference>

LOADW <picture reference>, <coordinate>

STOREW <picture reference>, <coordinate>

AREA <plane reference>, N. <index>

3.2.2.2 Sequence Control Instructions

GOTO <label>

unconditional jump to statement <label>

IFGO N. <index>, <relation>, <number reference>, <label>

A conditional jump to <label>, taken if N. <index> <relation><number reference> is true.

CALL label [{, <operand>}...]

A jump to subroutine with an optional argument list. Several actions occur as a result of executing this instruction:

1. All data lists except P are pushed into stacks.
2. The contents of RETADD (the return address) are pushed into the RETADD stack.
3. A sequence of implicit ASGN statements is executed, e.g.

```
ASGN    A.1,<operand 1>
ASGN    A.2,<operand 2>
            etc.
```

4. Control is given to the subroutine identified by <label>.

WAIT <number reference>

<number reference>is a count of the number of timer intervals to wait. If the operator strikes the CTL E key, control is returned immediately.

EXIT

1. All data lists except P are popped.
2. RETADD is saved for step 4.
3. RETADD is popped.
4. Control is returned to the statement at the location saved in step 2.

3.2.2.3 Data-Manipulating Instructions

INCR N. <index>, <number reference>
N. <index> is incremented by <number reference>

DECR N. <index>, <number reference>
N. <index> is decremented by <number reference>

ASGN <operand>, <operand> <*S12*>
The second <operand> is copied into the first <operand>.

RPIXEL N. <index>, <picture reference>, <coordinates>
The gray-scale value of the pixel at location <coordinates> in <picture reference> is loaded into N. <index>.

WPIXEL N. <index>, <picture reference>, <coordinates>
The integer in N. <index> is stored at <coordinates> in <picture reference> as a gray-scale value. If the binary representation of the value of N. <index> has more significant bits than the number specified in the DEFPIC for <picture reference>, low order bits are truncated.

3.2.2.4 Input-Output Instructions

SHOW <picture reference>

<picture reference> is displayed on the monitor. The display remains on the monitor until another picture I/O operation is executed. Control returns immediately to the program.

SHOWTL <picture reference>[,<picture reference>]

The pictures denoted by the operands are displayed on alternate scans on the monitor. A cursor is also displayed, which can be moved around on the display of the first <picture reference>, setting points to zero or all ones by pressing D or A, respectively. CTL I terminates execution of the instruction.

LOOK <picture reference>, <coordinates>
[,<number reference>]

This instruction may be used to cause a scanning device to display a picture on the monitor without reading the picture into core. Only the picture descriptor at <picture reference> is used; no core storage is used. In this way rasters larger than 128 x 128 pixels may be generated. The display remains on the monitor for <number reference> timer intervals. If the raster would fall outside the virtual raster for the scanning device, an error message is printed.

If the value of <number reference> is zero, one sweep of the raster only occurs. If <number reference> is omitted, the display is maintained until the operator types CTL R.

SCANM <picture reference>, <coordinates>

A LOOK instruction is executed twice using the operands:

picture reference , coordinates

Then a magnified 128 x 128 pixel window is superimposed on the previous display for one scan. These two displays alternate, generating a composite image of a high-resolution "magnifying glass" on a large, low resolution display. The "magnifying glass" may be moved over the large picture by typing T (up), F (left), H (right), space (down). The distance the "magnifying glass" moves in the X or Y direction at each step can be changed by typing X or Y respectively. The operator can then enter plus or minus signs. A plus sign doubles the previous movement distance and a minus halves it. Similarly, the scanner's sampling resolution in the X or Y direction and, hence the scope of the window can be changed by typing P or Q respectively. This is done by typing a plus or minus sign where plus doubles and a minus halves the sampling increment.

When the operator types CTL R, a final scan of the"magnifying glass" is made, and the window is read into core using a picture descriptor

and coordinates developed from parameters entered by the operator during processing of SCANM. The values of the parameters <picture reference> and <coordinates> are then changed to agree with the attributes of the window just read in.

SCAN <picture reference>, <coordinates>
The picture descriptor at <picture reference> is used to scan a window at location <coordinates> within a virtual raster of the current scanning device. If <picture reference> describes a window with either dimension larger than 128 pixels, that dimension is truncated to 128 pixels, and <picture reference> is changed to match. If a window is smaller than 128 x 128 pixels, it is stored as specified; i.e., variable size windows are accommodated.

PNABE <picture reference>, <coordinates>, <number reference
The gray-scale values of the pixels in a square neighborhood of side <number reference> centered at <coordinates> in <picture reference> are printed on the teletype. <number reference> must have a positive value less than 73.

DEFPC <picture reference>,< number reference$_1$>,
<number reference$_2$>, <number reference$_3$>,
<number reference$_4$>, <number reference$_5$>,
<number reference$_6$>

Defines a picture descriptor.
A DEFPIC must be executed before any reference is made to the operand < picture reference >. [1]

Operands:

<number reference$_1$> is the size in pixels of the window in the X direction.

<number reference$_2$> is the size in pixels of the window in the Y direction.

<number reference$_3$> is the scanner X step size (P). Must be non-negative and less than 8_{10}.

<number reference$_4$> is the scanner Y step size (Q). Must be non-negative and less than 8_{10}.

<number reference$_5$> is the magnification factor (H) to be used when displayingthe picture. Must be non-negative and less than 8_{10}.

<number reference$_6$> is the number of bits used to store the gray-scale value for each pixel.

RELPC <picture reference>

Releases all storage associated with <picture reference>.

---

1 When storing data into a picture, a reference can be made to an undefined picture. In this case, a definition is assumed:

DEFPIC <picture reference> , 128,128,0,0,0,4

Also if any of the operands are omitted in anyDEFPIC, they are assumed to take the above values, unless the picture was previously defined, in which case the old values are retained.

TRACE <picture reference>, C. <index>, N. <integer>
[,<increment reference>]

The value of N. <integer> is saved, and N. <integer> is set to one. A one-scan SHOW is executed using operand <picture reference>, alternating with a display of a small, bright point which acts as cursor. The cursor is moved around using commands similar to those for the SCANM command. Whenever S is typed, the coordinates of the cursor are placed in C. <index> and N. <integer> is incremented by one. In addition, if <increment reference> is present, the coordinates and gray-scale values of the neighbors of each sample point as indicated by increment reference are recorded on LINC tape. CTLI ends processing of the instruction, as does attempting to move the cursor off the picture. In addition, if N. <integer> exceeds the original value of the argument N. <integer>, processing ends.

SAVEPC <picture reference>

<picture reference> is written on LINC tape in a standard format.

LOADPC <picture reference>

A previously-saved picture is read from LINC tape and stored in <picture reference>. The descriptor for the saved picture is also loaded, performing an implicit DEFPIC.

TEXTI T. <index>

Characters typed on the console typewriter are stored in T. <index>

TEXTO <text reference>

Characters in <text reference> are typed on the console typewriter.

3.2.2.5 Data Storage Lists

All variables used as operands in S&T instructions are pre-declared and are stored in lists. Table 1 shows the eight different list types and the codes used to refer to them.

A variable in a list is accessed by means of an <index>, as in

```
SHOW P.3
```

which refers to the third picture descriptor. The following sequence could also be used to accomplish the same thing:

```
ASGN N.1,3
SHOW P.N. 1
```

The elements of two lists are themselves lists. Sublists of the C and I lists can be accessed as entities, or elements of the sublists can be accessed. The following statements assign a list of coordinates to the 5th sublist of list C, then changes the second coordinate pair of the list of coordinates just assigned.

```
ASGN  C.5( (19,3), (14,2), (6,25) )
ASGN  C.5.2, (2,14)
```

Sublists can be of variable length, as can the elements of lists B, F, and T. Coordinates and increments are stored as signed double-precision (23 bits and sign bit) integers, although there may be semantic restrictions on the magnitudes, (see section 3.2.4). Storage allocation and de-allocation is handled by the system, so sublists can have new values of different length assigned.

Each list can have only certain data types assigned to it. Assignands can be either lists or literals of the proper type, e.g.:

```
ASGN    B.1, #101101
ASGN    B.2, B.1
ASGN    T.1, B.2
TEXTC   T.1
ASGN    T.1, #1011100
TEXTO   T.1
```

The sequence above demonstrates the use of ASGN to print some bit strings on the teletype. Data type conversions are performed where necessary. Table 1 shows which combinations of assignee and assignand data types are legal. (In the first statement in the previous example, B.1 is the assignee and #101101 is the assignand.

A special <picture reference>, P.0, is used by PAXDRIVR to store pictures for use by PAX routines. A built-in DEFPIC, i.e.,

```
DEFPIC  P.0, 128,128,,,4,4
```

is in effect for this picture whenever PAX is in use.

TABLE 1 DATA STORAGE LISTS

CODE	LEGAL ASSIGNANDS	DESCRIPTION
A	Any	List of arguments used in call statement
B	A, T, B	List of bit-strings
C	A, T, C	List of coordinate lists
F	A, T, F	List of Polish post-fix Boolean functions
I	A, T, I	List of increment lists
N	A, T, N	List of double-precision integer storage cells
P	A, P	List of picture descriptors
T	Any except P	List of text strings

3.2.3 Operand Syntax

<bit-string reference> ::= B. <index>|<bit-string literal>

<direction reference> ::= <bit-string reference> <*S3*>

<plane reference> :: = <signed integer> <*S5*>

<picture reference> :: = P. <index>

<function reference> :: = F. <index> | <function literal>

<increment reference> :: = I. <index> |<increment literal>|
I. <index>, <index>

<coordinate reference> :: = C. <index> |<coordinate literal>|
C. <index>, <index>

<text reference> :: = T. <index> |<text literal>

<number reference> :: = N. <index> | <signed integer> <S*11*>

<bit-string literal> :: = #{1|0}...

<function literal> :: = <term string> <plane reference>

<term string >:: = {<elementary function> [{&|+|'}...];}...

<elementary function> :: = [<variable> [<operator> <variable>].

<variable> :: = ['] <plane reference> <direction number> <*S7*>

<direction number> :: = <integer> <*S6*>

<operator> :: = &|+

---

1 The lines under the brackets in this production indicate that these particular brackets are not metasymbols but that an elementary function is a string of variables and operators all enclosed in brackets.

<text literal> :: = ' <character >...'

<coordinate literal> :: =(<coordinates> [{, <coordinates>}...])

<coordinates> :: =(<number reference>, <number reference>) <*S8*>

<increment literal> :: =(<displacement> [{, <displacement>}...])

<displacement >:: = (<number reference>, <number reference>)<*S9*>

<signed integer> :: = [{+|-}] <integer>

<integer> :: = <digit>...

<digit> :: = 0|1|2|3|4|5|6|7|8|9

<relation> :: = .GT|.LT|.EQ|.NE|.GE|.LE

<label> :: = <alphabetic> [<alphanumeric>...] <*S10*>

<alphabetic> :: = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|
S|T|U|V|W|X|Y|Z

<alphanumeric> :: = <alphabetic>|<digit>

<character> :: = <any character on a teletype keyboard
except quote (')>

```
<operand> :: = <picture reference>
              | <function reference>
              | <increment reference>
              | <coordinate reference>
              | <text reference>
              | <number reference>
              | <bit-string reference>
              | <argument reference>

<index>   :: = <integer>| N. <integer>
```

3.2.4 Notes

<*S1*> <Bit-string reference> generates the availability list. Length must be 9.

<*S2*> <Bit-string reference> generates the ID byte. If present, length must be 9. If omitted, assumed value is #100000000.

<*S3*> <Bit-string reference> generates the direction list. Length must be 9.

<*S4*> If <plane reference> is omitted, it is assumed that the M-plane (plane number-1) has already been loaded with the operand plane.

<*S5*> <signed integer> must take one one of the values-1, 0, 1, ..., 7.

<*S6*> <integer> must take on one of the values 0, 1, 2, ...,8

<*S7*> Either: all of the <plane reference>s in an <elementary function> must be the same or all of the <direction number> s must be zero. Also, all <operator>s must be the same. (see reference 5 for explanation).

<*S8*> Each <number reference> must have a value such that $0 \leq$ value $< 2^{23} - 1$.

<*S9*> Each <number reference> must have a value such that $-7 \leq$ value ≤ 7.

<*S10*> <label> must have length no greater than six.

<*S11*> All <number references> must have a value such that -2^{23} <value $<2^{23}$.

<*S12*> See section 3.2.2.5 for restrictions on the values of <operand>.

4. INTERACTIVE PAX

4.1 Introduction

PAX II is a set of Fortran-callable subroutines which perform picture-processing operations on the elements of digitized images. (pixels)

The PAX II System is derived from a system developed at the University of Illinois to simulate the operations of the Illiac III's Pattern Articulation Unit.

An example of a PAX operation which is available is SLICE which identifies the pixels which have gray-scale values for which some condition, e.g., "greater than 8" is true. PAX operations can be used to remove noise from the picture, find edges and objects, and in general perform the pre-processing and feature extraction phases of the image processing task. For further details, the user is referred to the PAX II Programming Manual, reference [2].

PAX was intended to be used in a batch or off-line mode where pictures are stored on tape or disk, processed, and results printed, with no opportunity to monitor the process as it occurs. The S&T system extends PAX to include facilities which make it possible to use PAX interactively. That is, pictures can be selected, PAX routines called to process them, results examined on a video display, changes made in the sequence of execution of the PAX procedure and so forth, all from a console on-line to the computer.

Conversely, and in line with the two-part nature of the overall system objective (see section 1), special subroutines have been implemented which permit a PAX program running on the IBM 360 to control the acquisition of pictures directly.

4.2 Using Interactive PAX

All use of PAX from S&T is initiated via the S&T CALL instruction. Calls to PAX subroutines are of the form

CALL PAX, '<subroutine name>', <argument list>

where <subroutine name> can be any of the subroutines currently implemented. The operand, <argument list>, follows the argument list described for the PAX routine being called, with a few restrictions:

- all references to pictures or planes must be of the form P. <index>, where P. <index> was the operand of a DEFPIC instruction executed previously.
- Calls to PAX subroutines which include windows as arguments are replaced by calls to special versions of those routines which do not require window arguments. The names of these routines are the same as the standard PAX routines, except that an X precedes the subroutine name. If the addition of the X causes the name to be 7 characters long, the new name has the right hand character dropped. Thus CARDS becomes XCARDS and RANPIC becomes XRANPI. The new subroutines apply to the entire 128 x 128 picture instead of to a window.

Execution of CALL PAX, etc. causes the following to occur:

- Control is passed to the S&T built-in subroutine PAX (hereafter referred to as PAX_8), which acts as an interface between S&T and the 360 program PAXDRIVR. (See WK system command, section 3.2).
- PAX_8 moves the argument string from the CALL instruction to its buffer and sends it to the 360.
- PAXDRIVR reads the argument string, which includes the name of the subroutine to be called. The subroutine name is converted to EBCDIC and saved.

- All of the arguments are converted to the form which the PAX subroutine being called expects, i.e., integers are converted to fullwords, etc. Pointers are set up to each converted argument and saved in an arguments string to be passed in Register 1 to the PAX subroutine. Picture references point to arrays of PAX plane indices (stored in PAXDRIVR) which are initialized by DEFPIC instructions.
- When the argument string has been completed, the name of the subroutine to be called is used to enter a table of addresses. The address of the entry point of the subroutine is retrieved, register 1 is loaded with the address of the parameter list just constructed and control is transferred to the PAX subroutine. If the subroutine name is not found in the table, a 'command reject' reply is returned to S&T.
- If the PAX subroutines completes execution normally, a 'command completed OK' reply is sent to the 360, and PAXDRIVR lapses into the wait state until the next interrupt from the PDP8.
- As it is being executed, the PAX subroutine may require pictures or other data to be transferred to or from the PDP8/SMV. PAXDRIVR contains several subroutines which can be called from the PAX program for this purpose. Those subroutines are documented in section 4.3. The important point to know is that until a 'command completed OK' reply is sent to the PDP8, the PDP8 is enslaved to the 360, and may be called upon to perform various functions. Control is not returned to the S&T interpreter from PAX_8 until a 'command completed OK' reply is received.

4.3 Special PAX Subroutines

4.3.1 Introduction

A number of subprograms are provided as extensions to PAX for use with the S&T system. These routines are not available in the standard batch PAX, since S&T and the S-M-V may not be on-line when a batch job is run. However, these subroutines may be called from PAX programs, provided that such programs are executed only via S&T.

For example, to write a PAX program which performs a fairly large task semi-autonomously, the program can be coded in FORTRAN IV using PAX subroutines, e.g., Figure 5. This program can then be added to the library of PAX programs callable from S&T and to the subroutine name table in PAXDRIVR. It can then be invoked with an S&T statement (i.e., CALL PAX, 'SMART') and its progress can be observed on the monitor screen.

```
SUBROUTINE  SMART
DIMENSION, Etc.

          <set IXCOR, IXSS, IYCOR, IYSS to scan large portion of
          the subject at low resolution>.

          CALL SCAN (IXCOR, IXSS, IYCOR, IYSS)

          <interpret low-resolution image, compute coarse internal
          model of scene to direct further processing>.

          <scan smaller portions at higher resolution, update internal
          model of scene>.

          <display interpretation of scene for operator's evaluation>.

RETURN

END
```

Fig. 5 PAX PROGRAM : EXAMPLE OF AN AUTONOMOUS SUBROUTINE

4.3.2 Documentation of Special PAX Subroutines

Subroutine: PICIN

Purpose:

To transfer a picture from Illiac III core to a stack of PAX planes.

Usage:

CALL PICIN (IPO, NP)

Parameters:

IPO - Array of indices of planes into which picture is to be transferred.

NP - Number of planes in IPO.

Execution:

PICIN calls RDPIC to get a scan line from Illiac III core, then calls INROW to store it in IPO. This sequence occurs 128 times.

Subroutine: PICOUT

Purpose:

To transfer a picture from a stack of PAX planes to Illiac III core.

Usage:

CALL PICOUT (IPI, NP)

Parameters:

IPI - Array of indices of planes from which picture is to be transferred.

NP - Number of planes in IPI

Execution:

PICOUT calls OUTROW to get a row of IPI and then calls WRTPIC to transfer it to Illiac III core. This sequence occurs 128 times.

Subroutine RDPIC

Purpose:

To transfer one 128-element row of a 128 x 128 x 4-bit picture from the Illiac III core to an integer array.

Usage;

CALL RDPIC (NARRAY, & N).

Parameters:

NARRAY - 128-element, one-dimensional integer array. Must be of type INTEGER *4.

&N - N is statement number to which control is to be given when all 128 rows have been transferred.

Execution:

Each time RDPIC is called, one 128-element scan line of the picture currently in picture P.O. in Illiac III core is transferred to the 360 and stored in NARRAY.

On the 129th call, RDPIC returns control to statement N without transferring any data.

The next call to RDPIC reads the first scan line from the Illiac III, etc.

Subroutine: SCAN

Purpose:

To cause the Illiac III S-M-V system to read a picture into Illiac III core.

Usage:

SCAN (IXCOR, IYCOR, IXSS, IYSS,&N)

Parameter:

IXCOR - X coordinate of first point in window to be scanned. Must be of type INTEGER *4, and such that $0 \leq IXCOR \leq 2^{23} - 1$.

IXSS - X step size. Determines the sampling frequency along a scan line. Must be such that $0 \leq IXSS \leq 5$

IYCOR - Analogous to IXCOR

IYSS - Analogous to IYSS

N - Statement number to which control is to be returned if the desired window could not be scanned (see below).

Execution:

The arguments IXCOR, IYCOR, IXSS and IYSS are sent to the PDP-8 where they used to compute values for the S-M-V's parameter and coordinate words. For a complete discussion of the S-M-V, see reference [1].

A 128 x 128 x 4 - bit window is scanned into picture P.0 in Illiac III core, provided that the window falls wholly within the raster area of the device currently in use. If the window to be scanned would fall outside the raster area, one of two things occurs, depending on device type. If the device has a means of moving the object being scanned, as in the case of a program-controlled microscope stage, the object is shifted so that the desired window falls within the raster area. In this way, the entire object area is treated as a virtual raster, so that a maximum of 2^{23} location are addressable in each direction. Note, however, that injudicious use of this facility can result in long wait times while mechanical motion is effected.

The parameters IXSS and IYSS are inserted into the sampling increment slots (P and Q) of the S-M-V Parameter word. These parameters determine how frequently the scanner will measure the gray-scale value of the image. For example, if IXSS=3, then of the locations along the scan line where the scanner is capable of making a measurement, only one in eight (2^3) will actually be made. The effect is to control the coarseness of the digitized image. Since the number of sample points is fixed at 128 per scan line for this subroutine, the effect is also to control the size of the image area being read in. The S-M-V hardware requires that the locations of all the points be integral multiples of the sample spacing. This means that if IXCOR and IYCOR are not integral multiples of 2**IXSS and 2**IYSS, respectively,then their binary representation will be truncated so that they are. For example, if IXSS=2 and IXCOR=154, then IXCOR becomes 152.

Subroutine: WRTPIC

Purpose:

To transfer one 128-element row of a 128 x 128 x 4-bit picture from an integer array to Illiac III core.

Usage:

CALL WRTPIC (NARRAY, &N)

Parameters:

NARRAY - 128- element ,one-dimensional integer array. Must be of type INTEGER * 4.

N - N is statement number to which control is to be given when 128 rows have been transferred.

Execution:

Each time WRTPIC is called, the 128 elements of NARRAY are transferred to Illiac III core and stored in picture P.0.

Successive calls to WRTPIC transfer successive scan lines to the Illiac III. On the 129th call, WRTPIC returns control to statement number N without transferring any data. The next call transfers NARRAY to the first scan line in P.O., etc.

6. Error Message Format

Error messages are of the form

* E <integer> [<text>]

To interpret the error message, use <integer> as an index to a table of error messages. Entries in the table explain the cause of the error condition and recommend remedial action. The optional<text> entry provides detailed information about the error condition to aid in debugging.

REFERENCE

[1] Dunn,L.,et.al; "Parametric Description of a Scan-display System," Proc. 1969 Spring Joint Computer Conference pp.187-206

[2] Borovec, R., "The PAX II Picture Processing System-Programming Manual," University of Illinois, Dept. of Computer Science Report No. 314, March 1969

[3] McCormick, B., et.al; "Illiac III Programming Manual," University of Illinois, Dept. of Computer Science.

[4] Digital Equipment Corporation "PDP-8 Disk Monitor Manual."

[5] McCormick, B., et.al.; "The Pattern Articulation Unit of the Illiac III : Homogeneous Boolean Functions in the Iterative Array." University of Illinois, Dept. of Computer Science Report No. 253.

[6] International Business Machines Corp. "PL/1 Language Specifications," IBM Systems Reference Library, Form C28-6571

APPENDIX:

Table of System Commmands and Show-and-Tell Language Statements

System Commands

Code	Function
CH	Change program statements
DL	Delete program statements
DV	Define scanning device
GO	Execute program in the ICA
IN	Insert program statements
KL	Kill PAXDRIVR
LD	Load a previously-translated program
LF	Log off
LØ	Log on
LS	List storage
SV	Save ICA
WK	Test for 360 awake
XG	Translate and go
XL	Translate

Show-and-Tell Language Statements

PAU Instructions

Code	Function
AREA	Count ones in a plane
BOOLE	Evaluate homogeneous Boolean Function
CLEARP	Set plane to zeros
CONNEC	Determine graph connectivity
COPY	Copy plane
COPYC	Complement plane and copy
LIST	Scan plane and store coordinates of ones
LOADW	Load window into Iterative Array
PLAND	Perform logical AND between two planes

PLEXOR	Perform logical EXCLUSIVE OR between two planes
PLOR	Perform logical OR between two planes
SETP	Set planes to ones
SHIFT	Shift plane
STOREW	Store window from Iterative Array
TALLY	Base-1 count up
TALLYH	Base-1 count down

Sequence Control Instruction

Code	Function
CALL	Subroutine call
EXIT	Return from subroutine
GOTO	Unconditional jump
IFGO	Conditional jump
WAIT	Wait for operator response or timeout

Data-manipulating Instructions

Code	Function
ASGN	Assign a value to a data item
DECR	Decrement a counter
INCR	Increment a counter
RPIXEL	Read picture element
WPIXEL	Write picture element

Input-Output Instructions

Code	Function
DEFPC	Define picture
LOADPC	Read picture from LINC tape
LOOK	Scan picture without input to core
PNABE	Type gray-scale values of a small region
RELPC	Release picture storage
SAVEPC	Write picture on LINC tape

SCAN	Select picture and read into core (program control)
SCANM	Select picture and read into core (operator control)
SHOW	Display picture on monitor
SHOWTL	Display picture with operator feedback
TEXTI	Read text from teletype
TEXTØ	Write text on teletype
TRACE	Record cursor movement and sample neighborhoods

orm AEC-427
(6/68)
AECM 3201

U.S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.	2. TITLE
429 COO-2118-0003	SHOW-AND-TELL An Interactive Programming System For Image Processing

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific end technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

John S. Read, Research Programmer

Orgenization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signeture	Date
John S. Read	February 18, 1971

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC petent group for cleerence.
- [ ] c. Patent clearence not required.